Table of Contents

Summary

FreeBSD-101-Hacks

A collection of FreeBSD hacks.

Project Homepage

https://github.com/NanXiao/FreeBSD-101-Hacks

Get system version

There are `2` versions of `FreeBSD`: kernel and userland. Use "`freebsd-version -ku`" can output both versions:

```
# freebsd-version -uk
10.3-RELEASE-p4
10.3-RELEASE-p5
```

The first one is kernel version info. You can see the versions of both kernel and userland is `10.3`, but kernel patch level is `p4`, while userland is `p5`. That's because in some cases, only the userland requires a patch whilst kernel not, so kernel patch level doesn't change. If there is no option, "`freebsd-version`" prints userland version only:

```
# freebsd-version
10.3-RELEASE-p5
```

"`uname -r`" command also outputs the kernel version:

```
# uname -r
10.3-RELEASE-p4
```

Reference:
Why the outputs of “freebsd-version” and “freebsd-version -k” are different?.

Enable root login over SSH

By default, `FreeBSD` box doesn't allow `root` user login over `SSH`, to enable it, you need to modify `SSH` daemon configuration file (`/etc/ssh/sshd_config`).

Change the following line:

```
#PermitRootLogin no
```

to:

```
PermitRootLogin yes
```

Then restart `SSH` daemon:

```
# /etc/rc.d/sshd restart
```

Reference:
[How to enable root login over SSH on FreeBSD 8.2](#).

Configure proxy

If your `FreeBSD` box works behind a proxy, you may need to configure proxy to make it access network.

For `csh` or `tcsh`, set proxy in `/etc/csh.cshrc`:

```
setenv HTTP_PROXY http://web-proxy.xxxxxx.com:8080
setenv HTTPS_PROXY https://web-proxy.xxxxxx.com:8080
```

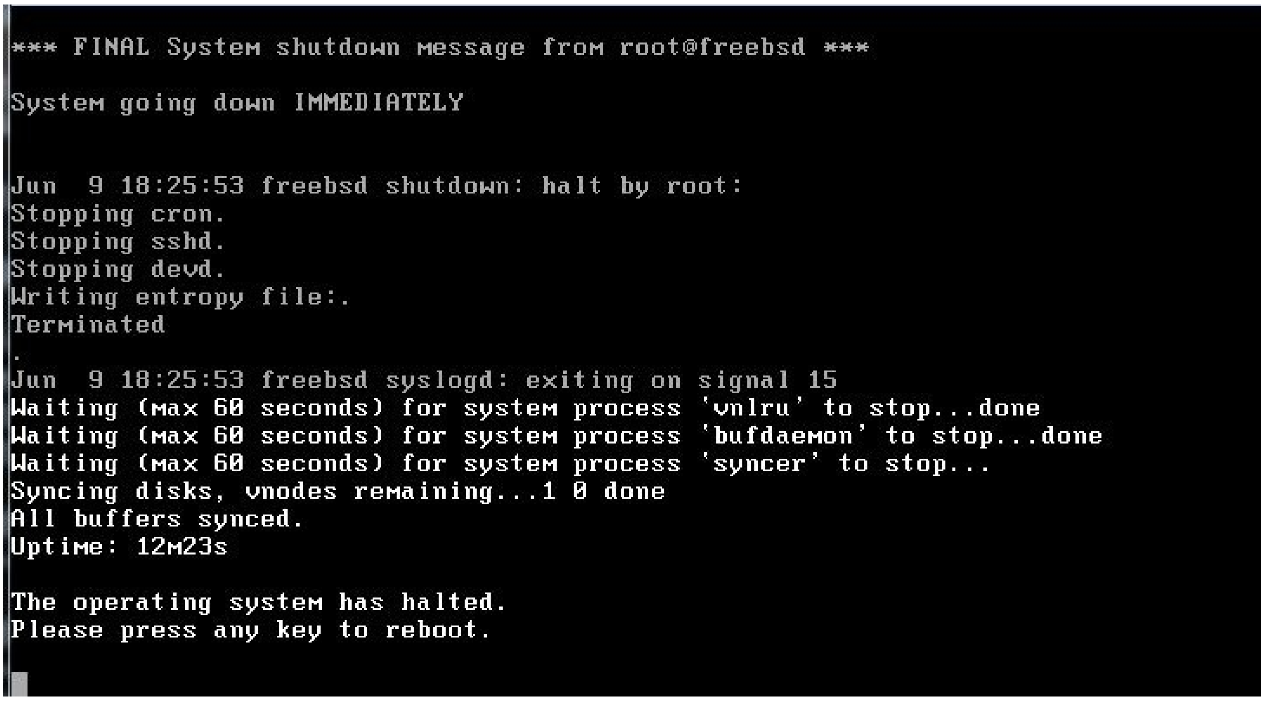
For `sh`, set proxy in `/etc/profile`:

```
export HTTP_PROXY http://web-proxy.xxxxxx.com:8080
export HTTPS_PROXY https://web-proxy.xxxxxx.com:8080
```

Reference:
Using FreeBSD inside a controlled network – A required HTTP Proxy and No FTP.

Shutdown

Compared to some `GNU/Linux` flavors, " `shutdown -h now` " command just halts the `FreeBSD` , not turns off the power. Refer the following screen shot from my Virtual Machine:



```
*** FINAL System shutdown message from root@freebsd ***

System going down IMMEDIATELY


Jun  9 18:25:53 freebsd shutdown: halt by root:
Stopping cron.
Stopping sshd.
Stopping devd.
Writing entropy file:.
Terminated
.
Jun  9 18:25:53 freebsd syslogd: exiting on signal 15
Waiting (max 60 seconds) for system process 'vnlru' to stop...done
Waiting (max 60 seconds) for system process 'bufdaemon' to stop...done
Waiting (max 60 seconds) for system process 'syncer' to stop...
Syncing disks, vnodes remaining...1 0 done
All buffers synced.
Uptime: 12m23s

The operating system has halted.
Please press any key to reboot.
```

You can see, after running " `shutdown -h now` ", the system is just halted, and press any key can reboot the system immediately.

If you want to halt and power off the system, you should execute " `shutdown -p now` ".

Reference:
SHUTDOWN.

Reboot

If you want to reboot `FreeBSD` immediately, you can use the following methods:

(1) Run " `reboot` " command directly:

```
# reboot
```

(2) Utilize " `shutdown` " command:

```
# shutdown -r now
```

" `-r` " option stands for `reboot` .

Upgrade system

You should make use of "`freebsd-update`" utility to upgrade your system non-periodically to keep your system safe and not stale. E.g., after installing a fresh `10.3` version:

```
# freebsd-version -ku
10.3-RELEASE
10.3-RELEASE
```

You should update the patches released by `FreeBSD` team:

```
# freebsd-update fetch install
```

When it is done, check the system version again:

```
# freebsd-version -ku
10.3-RELEASE-p4
10.3-RELEASE-p5
```

Yeap! your system is refreshed now!

References:
FreeBSD Man Pages;
What are the first commands I run after installing FreeBSD? Or How to patch FreeBSD? Or How to install ports on FreeBSD?.

Upgrade to new release

This article takes upgrading system from `10.2-RELEASE` to `10.3-RELEASE` as an example, but you must notice this tutorial may **not** apply to other versions. So please read the related release notes.

(1) Check current version:

```
# freebsd-version -ku
10.2-RELEASE
10.2-RELEASE
```

(2) Run " `freebsd-update upgrade` " command to upgrade current system to a new version:

```
# freebsd-update upgrade -r 10.3-RELEASE
......
To install the downloaded upgrades, run "/usr/sbin/freebsd-updat
e install".
```

(3) The last log of " `freebsd-update upgrade -r 10.3-RELEASE` " prompts executing " `freebsd-update install` ":

```
# freebsd-update install
src component not installed, skipped
Installing updates...
Kernel updates have been installed.  Please reboot and run
"/usr/sbin/freebsd-update install" again to finish installing up
dates.
```

(4) Reboot the system and execute " `freebsd-update install` " again:

```
# reboot
# freebsd-update install
src component not installed, skipped
Installing updates... done.
```

(5) Now check the version:

```
# freebsd-version -ku
10.3-RELEASE-p4
10.3-RELEASE-p5
```

The system has been upgraded to `10.3-RELEASE` now.

Reference:
freebsd-update;
FreeBSD 10.3-RELEASE Installation Instructions.

Install software

If you don't have any specific requirements for needed software, you can use `pkg` utility to install the pre-built package on `FreeBSD`, and it looks like using `yum` or `apt-get` on `GNU/Linux` boxes. E.g., install `python` on your host:

```
# pkg install python
......
===========================================================================

Note that some standard Python modules are provided as separate ports
as they require additional dependencies. They are available as:

bsddb           databases/py-bsddb
gdbm            databases/py-gdbm
sqlite3         databases/py-sqlite3
tkinter         x11-toolkits/py-tkinter

===========================================================================
```

Now you can run `python` now:

```
# python
Python 2.7.11 (default, Jun  5 2016, 01:23:11)
[GCC 4.2.1 Compatible FreeBSD Clang 3.4.1 (tags/RELEASE_34/dot1-final 208032)] on freebsd10
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

But the end logs of installing `python` prompts we still need to install other `4` packages: `py-bsddb`, `py-gdbm`, `py-sqlite3` and `py-tkinter`, otherwise "`import sqlite3`" will complain:

```
>>> import sqlite3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/local/lib/python2.7/sqlite3/__init__.py", line 24,
in <module>
    from dbapi2 import *
  File "/usr/local/lib/python2.7/sqlite3/dbapi2.py", line 28, in
 <module>
    from _sqlite3 import *
ImportError: No module named _sqlite3
```

This introduces another installation method: using ports. This method will download, compile and install from source code, and the benefit of this approach is you can do some customizations if you want. You can find all ports under `/usr/ports` directory. Now you begin to install `py-bsddb` (follow this method to set up `py-gdbm` , `py-sqlite3` and `py-tkinter`):

```
# cd /usr/ports/databases/py-bsddb
# make install clean
```

After all are installed, we can use `sqlite` in `python` now:

```
# python
Python 2.7.11 (default, Jun  5 2016, 01:23:11)
[GCC 4.2.1 Compatible FreeBSD Clang 3.4.1 (tags/RELEASE_34/dot1-
final 208032)] on freebsd10
Type "help", "copyright", "credits" or "license" for more inform
ation.
>>> import sqlite3
```

The other method is installing from `DVD` :
(1) Put `DVD` in the drive;
(2) Mount `DVD` :

```
mkdir -p /dist
mount -t cd9660 /dev/cd0 /dist
```

(3) Install the package (E.g., `xorg`):

```
env REPOS_DIR=/dist/packages/repos pkg install xorg
```

References:
Packages and Ports: Adding Software in FreeBSD;
Get started with FreeBSD: A brief intro for Linux users;
FreeBSD 10 install packages from DVD.

Search software

If you want to search a specified application, you can use " `pkg search` " command. For example:

```
# pkg search lsof
lsof-4.90.b,8                  Lists information about open file
s (similar to fstat(1))
p5-Unix-Lsof-0.0.5_2           Unix::Lsof -- a wrapper to the Un
ix lsof utility
```

If you want to know the software's path in the ports tree, you can use `-o` option:

```
# pkg search -o lsof
sysutils/lsof                  Lists information about open file
s (similar to fstat(1))
sysutils/p5-Unix-Lsof          Unix::Lsof -- a wrapper to the Un
ix lsof utility
```

In case the Ports Collection is already installed, you can employ `whereis` command:

```
# whereis lsof
lsof: /usr/local/sbin/lsof /usr/local/man/man8/lsof.8.gz /usr/po
rts/sysutils/lsof
```

Another tricky method is using " `echo` " command:

```
# echo /usr/ports/*/*lsof*
/usr/ports/distfiles/lsof_4.90B.freebsd.tar.bz2 /usr/ports/sysut
ils/lsof
```

Reference:
[Finding Software](#).

Remove software

After installing application, you can use " `pkg delete` " command to remove it:

```
# pkg delete lsof
Checking integrity... done (0 conflicting)
Deinstallation has been requested for the following 1 packages (
of 0 packages in the universe):

Installed packages to be REMOVED:
        lsof-4.90.b,8

Proceed with deinstalling packages? [y/N]: y
[1/1] Deinstalling lsof-4.90.b,8...
[1/1] Deleting files for lsof-4.90.b,8: 100%
```

You can also enter the port directory, and execute " `make deinstall` " instruction:

```
# pwd
/usr/ports/sysutils/lsof
# make deinstall
===>  Deinstalling for lsof
===>   Deinstalling lsof-4.90.b,8
Updating database digests format: 100%
Checking integrity... done (0 conflicting)
Deinstallation has been requested for the following 1 packages (
of 0 packages in the universe):

Installed packages to be REMOVED:
        lsof-4.90.b,8
[1/1] Deinstalling lsof-4.90.b,8...
[1/1] Deleting files for lsof-4.90.b,8: 100%
```

Reference:
Using the Ports Collection.

Install source code

Download `FreeBSD` source code according architecture, version:

```
fetch ftp://ftp.freebsd.org/pub/`uname -s`/releases/`uname -m`/`
uname -r | cut -d'-' -f1,2`/src.txz
```

Uncompress the source package:

```
tar -C / -xvzf src.txz
```

Make sure the `src` is included in `Components` line in `/etc/freebsd-update.conf`, so when using `freebsd-update` to update system, also the source code is refreshed.

```
# cat /etc/freebsd-update.conf
......
# Components of the base system which should be kept updated.
Components src world kernel
......
```

References:
A question about downloading FreeBSD kernel code;
Install FreeBSD kernel source after installed freebsd.

Build kernel

If you want to build a custom kernel yourself, you need to install the source code firstly (you can take post as a reference). Then enter the configuration directory of your machine:

```
# cd /usr/src/sys/`uname -m`/conf
```

Copy a new kernel configuration file:

```
# cp GENERIC NAN_FIRST_BUILD
```

You can modify the new configuration file (`NAN_FIRST_BUILD`) according to your needs.

Build the kernel:

```
# cd /usr/src
# make buildkernel KERNCONF="NAN_FIRST_BUILD"
```

Once it is finished, install the fresh kernel binary and reboot:

```
# make installkernel KERNCONF="NAN_FIRST_BUILD"
# reboot
```

Check the kernel version, you will find now the OS is harnessing your newly built kernel now:

```
# uname -a
FreeBSD FreeBSD 10.3-RELEASE-p5 FreeBSD 10.3-RELEASE-p5 #0: Tue
Jul 19 17:52:57 CST 2016     root@FreeBSD:/usr/obj/usr/src/sys/N
AN_FIRST_BUILD  amd64
```

Reference:

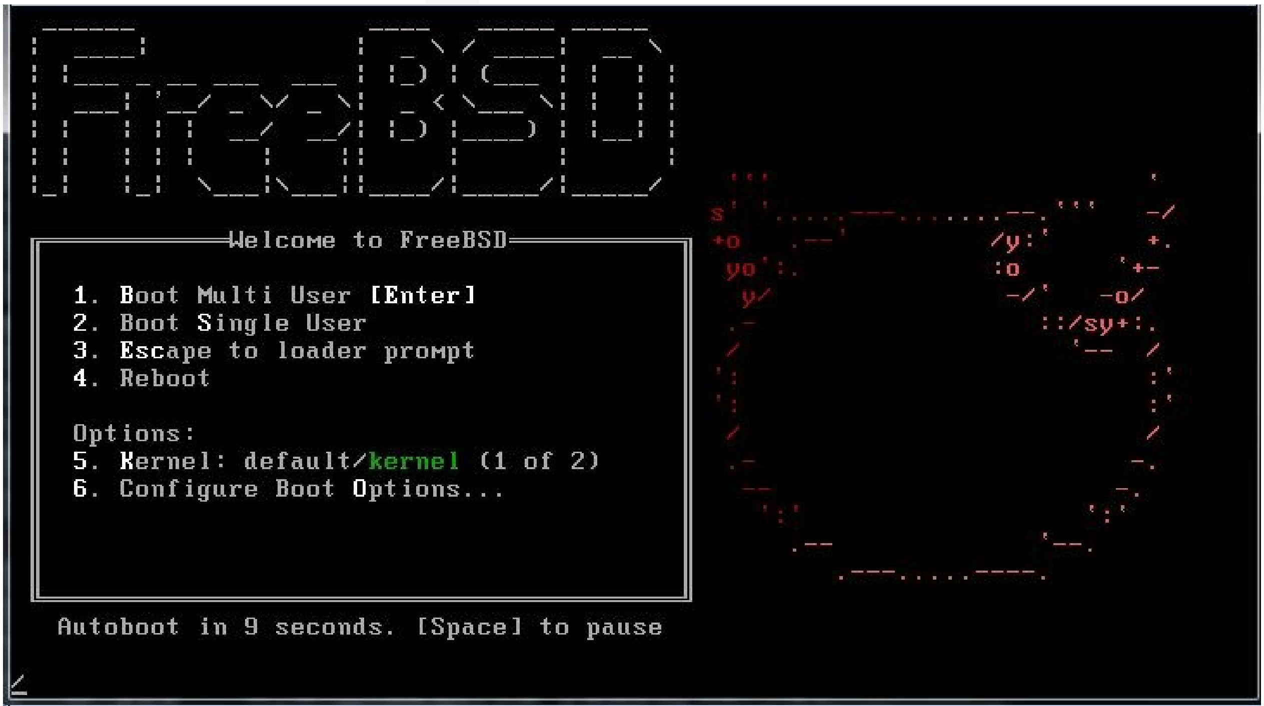
How to build a custom kernel in FreeBSD.

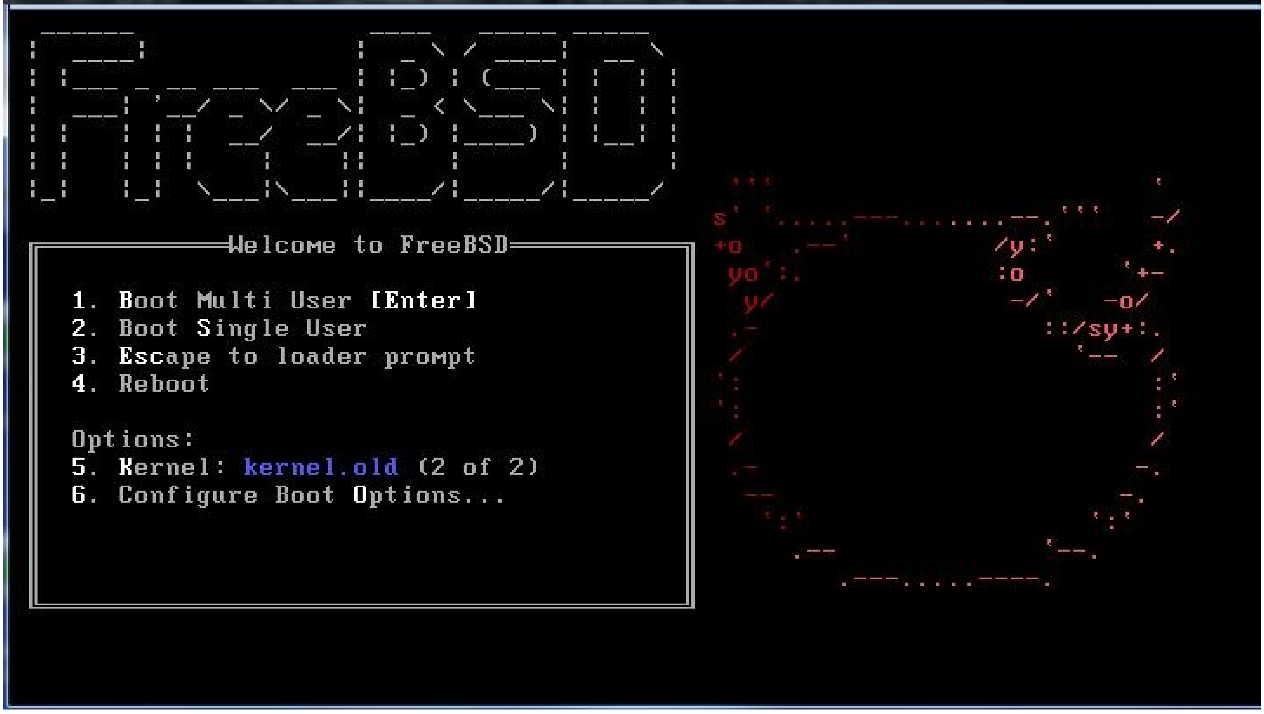
Backup old kernel

Every time, when a new kernel is installed, the last installed kernel will be backed up in a directory who is called `kernel.old`. So it is important to remember to save a copy of usable kernel in case the newly-built kernel can't work. For example:

```
# mv /boot/kernel.old /boot/kernel.good
# mv /boot/kernel /boot/kernel.bad
# mv /boot/kernel.good /boot/kernel
```

During boot, you can press `5` to select which kernel to run:





References:

FreeBSD: reverting to previous kernel;

If Something Goes Wrong.

CURRENT VS STABLE

`FreeBSD` has two development branches: `FreeBSD-CURRENT` and `FreeBSD-STABLE`.

`FreeBSD-CURRENT` is the active development branch of `FreeBSD`, and all new development work, such as adding new feature, will occur in `FreeBSD-CURRENT` branch firstly.

`FreeBSD-STABLE` is span off from `FreeBSD-CURRENT`, and for production version. It mainly accepts bug and security fixes.

You should notice, branches are not points but constant streams of development. And the `FreeBSD RELEASE` version is a point, and it is just a snapshot of `FreeBSD-STABLE` branch at some time.

References:
Tracking a Development Branch;
Release Engineering.

Update Ports Collection

The stuff under `/usr/ports` is referred as Ports Collection, and we can employ `portsnap` command to update it:

(1) If it is the first time to use `portsnap`, you should run following command:

```
# portsnap fetch extract
```

Otherwise, you can encounter errors as shown here:

```
......
/usr/ports was not created by portsnap.
You must run 'portsnap extract' before running 'portsnap update'
.
```

(2) Hereafter, you only need to execute "`portsnap fetch update`" to update the Ports Collection.

Reference:
Using the Ports Collection;
A question about using portsnanp to upgrade ports tree.

Accept default options during installing port

During building and installing port, if you are tired of the prompt, and want to use all default options, you can utilize `BATCH` :

```
# make install clean BATCH=yes
```

Reference:
How can I avoid the prompts when installing a FreeBSD port?.

Enable ls colour output

`ls -G` can make `ls` output colourful. If you use `csh`, you can add following statement in `.cshrc`:

```
alias ls        ls -G
```

Reference:
FreeBSD: Enable Colorized ls Output.

Clear directory structure

`FreeBSD` has a clear separation between base operating system and user-added applications, that means everything which does not belong to the base operating system is under `/usr/local` directory, and the `/usr/local` directory contains a directory structure that mostly mirrors the structure found in the `/` or `/usr` directory.

For example, the application installed from port tree or `package` command will be located in `/usr/local/bin` or `/usr/local/sbin`, corresponding to `/bin`, `/sbin`, `/usr/bin` and `/usr/local/sbin`; and `/etc` directory contains configuration files for the base operating system while `/usr/local/etc` for user-added applications.

References:
A Comparative Introduction to FreeBSD for Linux Users;
Directory Structure;
Ten Things I like About FreeBSD.

Change the shell

If you were previously working on `Linux`, you may miss `Bash`, and not accustomed to `csh` which is shipped on `FreeBSD` by default. You can change the shell to `Bash` following the below instructions:

(1) The `Bash` must already be installed on your `FreeBSD`, and its full path should also be included in `/etc/shells`. E.g:

```
 # pkg install bash
Updating FreeBSD repository catalogue...
FreeBSD repository is up-to-date.
All repositories are up-to-date.
The following 1 package(s) will be affected (of 0 checked):

New packages to be INSTALLED:
        bash: 4.3.42_1
......
```

Then you will find the `Bash` path is presented in `/etc/shells` automatically:

```
 # cat /etc/shells
# $FreeBSD: releng/10.3/etc/shells 59717 2000-04-27 21:58:46Z ac
he $
#
# List of acceptable shells for chpass(1).
# Ftpd will not allow users to connect who are not using
# one of these shells.

/bin/sh
/bin/csh
/bin/tcsh
/usr/local/bin/bash
/usr/local/bin/rbash
```

(2) Execute `chsh -s /usr/local/bin/bash`:

```
# chsh -s /usr/local/bin/bash
chsh: user information updated
```

After you login again, you can find the shell is `bash` now:

```
# echo $SHELL
/usr/local/bin/bash
```

Reference:
Shells.

Change PATH environment variable

In `csh`, if you want to change `$PATH` environment variable, you should modify `.cshrc` file:

```
# cat .cshrc
......
set path = (/sbin /bin /usr/sbin /usr/bin /usr/games /usr/local/
sbin /usr/local/bin $HOME/bin $HOME/gocode/bin)
......
```

For example, you add "`/usr/local/go/bin`" into `$PATH`:

```
# cat .cshrc
......
set path = (/sbin /bin /usr/sbin /usr/bin /usr/games /usr/local/
sbin /usr/local/bin $HOME/bin $HOME/gocode/bin /usr/local/go/bin
)
......
```

Then executing "`source .cshrc`" command, you will find the new `$PATH` takes effect:

```
# source .cshrc
# echo $PATH
/sbin:/bin:/usr/sbin:/usr/bin:/usr/games:/usr/local/sbin:/usr/lo
cal/bin:/root/bin:/root/gocode/bin:/usr/local/go/bin
```

Reference:
Misc Tips & Tricks.

Colorize bash

(1) In your `$HOME` directory, create `..dir_colors` file:

```
# Below, there should be one TERM entry for each termtype that i
s colorizable
TERM ansi
TERM color-xterm
TERM con132x25
TERM con132x30
TERM con132x43
TERM con132x60
TERM con80x25
TERM con80x28
TERM con80x30
TERM con80x43
TERM con80x50
TERM con80x60
TERM cons25
TERM console
TERM cygwin
TERM dtterm
TERM Eterm
TERM gnome
TERM konsole
TERM kterm
TERM linux
TERM linux-c
TERM mach-color
TERM putty
TERM rxvt
TERM rxvt-cygwin
TERM rxvt-cygwin-native
TERM rxvt-unicode
TERM screen
TERM screen-bce
TERM screen-w
TERM screen.linux
```

```
TERM vt100
TERM xterm
TERM xterm-256color
TERM xterm-color
TERM xterm-debian

# Below are the color init strings for the basic file types. A c
olor init
# string consists of one or more of the following numeric codes:
# Attribute codes:
# 00=none 01=bold 04=underscore 05=blink 07=reverse 08=concealed
# Text color codes:
# 30=black 31=red 32=green 33=yellow 34=blue 35=magenta 36=cyan
37=white
# Background color codes:
# 40=black 41=red 42=green 43=yellow 44=blue 45=magenta 46=cyan
47=white
NORMAL 00           # global default, although everything should
 be something.
FILE 00             # normal file
DIR 01;34           # directory
LINK 01;36          # symbolic link.  (If you set this to 'targe
t' instead of a
                    # numerical value, the color will match the
file pointed to)
FIFO 40;33          # pipe
SOCK 01;35          # socket
DOOR 01;35          # door
BLK 40;33;01        # block device driver
CHR 40;33;01        # character device driver
ORPHAN 01;05;37;41  # orphaned syminks
MISSING 01;05;37;41 # ... and the files they point to

# This is for files with execute permission:
EXEC 01;32

# List any file extensions like '.gz' or '.tar' that you would l
ike ls
# to colorize below. Put the extension, a space, and the color i
nit string.
```

```
# (and any comments you want to add after a '#')

.cmd 01;32 # executables (bright green)
.exe 01;32
.com 01;32
.btm 01;32
.bat 01;32
.sh  01;32
.csh 01;32

.tar 01;31 # archives / compressed (bright red)
.tgz 01;31
.arj 01;31
.taz 01;31
.lzh 01;31
.zip 01;31
.z   01;31
.Z   01;31
.gz  01;31
.bz2 01;31
.bz  01;31
.tbz2 01;31
.tz  01;31
.deb 01;31
.rpm 01;31
.rar 01;31        # app-arch/rar
.ace 01;31        # app-arch/unace
.zoo 01;31        # app-arch/zoo
.cpio 01;31       # app-arch/cpio
.7z  01;31        # app-arch/p7zip
.rz  01;31        # app-arch/rzip

.jpg 01;35 # image formats
.jpeg 01;35
.gif 01;35
.bmp 01;35
.ppm 01;35
.tga 01;35
.xbm 01;35
.xpm 01;35
```

```
.tif 01;35
.tiff 01;35
.png 01;35
.mng 01;35
.xcf 01;35
.pcx 01;35
.mpg 01;35
.mpeg 01;35
.m2v 01;35  # MPEG-2 Video only
.avi 01;35
.mkv 01;35  # Matroska (http://matroska.org/)
.ogm 01;35  # Ogg Media File
.mp4 01;35  # "Offical" container for MPEG-4
.m4v 01;35  # MPEG-4 Video only
.mp4v 01;35 # MPEG-4 Video only
.mov 01;35  # Quicktime (http://developer.apple.com/qa/qtw/qtw99
.html)
.qt 01;35   # Quicktime (http://developer.apple.com/qa/qtw/qtw99
.html)
.wmv 01;35  # Windows Media Video
.asf 01;35  # Advanced Systems Format (contains Windows Media Vi
deo)
.rm 01;35   # Real Media
.rmvb 01;35 # Real Media Variable Bitrate
.flc 01;35  # AutoDesk Animator
.fli 01;35  # AutoDesk Animator
.gl 01;35
.dl 01;35

.pdf 00;32 # Document files
.ps 00;32
.txt 00;32
.patch 00;32
.diff 00;32
.log 00;32
.tex 00;32
.doc 00;32

.mp3 00;36 # Audio files
.wav 00;36
```

```
.mid 00;36
.midi 00;36
.au 00;36
.ogg 00;36
.flac 00;36
.aac 00;36
```

(2) Add following statements in `/etc/profile` :

```
eval $(dircolors -b ~/.dir_colors)

if [[ ${EUID} == 0 ]] ; then
    PS1='\[\033[01;31m\]\h\[\033[01;34m\] \W \$\[\033[00m\] '
else
    PS1='\[\033[01;32m\]\u@\h\[\033[01;34m\] \w \$\[\033[00m\] '
fi

CLICOLOR="YES";    export CLICOLOR
LSCOLORS="ExGxFxdxCxDxDxhbadExEx";    export LSCOLORS
```

Reference:
Colorize your bash like linux..

Install bash-completion

bash-completion is a really handy tool which can help you auto-complete many commands, such as `git`. To use it, you need `2` steps:

(1) Install it:

```
pkg install bash-completion
```

(2) Add following command in your `~/.profile`:

```
[[ $PS1 && -f /usr/local/share/bash-completion/bash_completion.sh ]] && \
        source /usr/local/share/bash-completion/bash_completion.sh
```

Rehash command

After removing the application in the directory listed in the `$PATH` variable, you may even see the command prompt when you type the first characters:

```
# pkg delete lsof
Checking integrity... done (0 conflicting)
Deinstallation has been requested for the following 1 packages (
of 0 packages in the universe):

Installed packages to be REMOVED:
        lsof-4.90.b,8

Proceed with deinstalling packages? [y/N]: y
[1/1] Deinstalling lsof-4.90.b,8...
[1/1] Deleting files for lsof-4.90.b,8: 100%
# ls
ls        ls-F      lsextattr lsof      lspci     lsvfs
```

While running it will prompt " `Command not found.` " definitely:

```
# lsof
lsof: Command not found.
```

Np matter install or remove application , you need to run `rehash` command in `csh` to causes the shell to recompute a table of where commands are located.

Reference:
Misc Tips & Tricks;
Rehash.

Search backwards in csh

In `bash`, `Ctrl + R` provides the handy "search backwards" feature which isn't shipped by `csh` by default. If you want to use the similar function in `csh`, you can employ `csh` built-in `bindkey` command to implement it:

```
# cat .cshrc
......

bindkey "^R" i-search-back
```

Then you can use "search backwards" when pressing `Ctrl + R`:

```
root@FreeBSD:~ # pkg install subversion
bck:pk
```

References:
Ctrl-R to search backwards for shell commands in csh.

Trim a file in csh

In `bash`, if you want to clear out a file, the easiest command may be "`> file`":

```
# cat test.txt
Hello world!
# > test.txt
# cat test.txt
#
```

But in `csh`, "`> file`" will cause "`Invalid null command.`" error:

```
# cat test.txt
Hello world!
# > test.txt
Invalid null command.
```

The solution is using "`: > file`" command:

```
# cat test.txt
Hello world!
# : > test.txt
# cat test.txt
#
```

Or employ "`cat /dev/null > file`".

Reference:
What does a leading colon (:) mean in a script?.

/var/run/dmesg.boot

`/var/run/dmesg.boot` file contains the `FreeBSD` boot messages, and this is very helpful for checking hardware related info:

```
# cat /var/run/dmesg.boot | grep -i CPU
CPU: Intel(R) Core(TM) i5-4310U CPU @ 2.00GHz (2594.03-MHz K8-cl
ass CPU)
cpu0: <ACPI CPU> on acpi0
```

Because the capacity of system message buffer is limited, `FreeBSD` employs `/var/run/dmesg.boot` in case the content of buffer is flushed.

References:
Devices and Device Nodes;
Back To Basics: Unix Differences in Performing Tasks.

Display swap space utilization

On `FreeBSD`, you can use `pstat -s` to display swap space utilization:

```
# pstat -s
Device          1K-blocks     Used    Avail Capacity
/dev/da0p3        1048540        0  1048540     0%
```

There is another specified `swapinfo` command you can use:

```
# swapinfo
Device          1K-blocks     Used    Avail Capacity
/dev/da0p3        1048540        0  1048540     0%
```

You can also display swap info in "Human-readable" format:

```
# swapinfo -h
Device          1K-blocks     Used    Avail Capacity
/dev/da0p3        1048540       0B     1.0G     0%
```

Or display swap space size in megabyte unit:

```
# swapinfo -m
Device          1M-blocks     Used    Avail Capacity
/dev/da0p3           1023        0     1023     0%
```

References:
PSTAT.

Use "ps" to display process status

If you run " `ps` " command without any options, it will only show **your** processes who have a controlling terminal:

```
# ps
PID TT  STAT    TIME COMMAND
854 v0  Is+  0:00.00 /usr/libexec/getty Pc ttyv0
710 v1  Is+  0:00.00 /usr/libexec/getty Pc ttyv1
711 v2  Is+  0:00.00 /usr/libexec/getty Pc ttyv2
712 v3  Is+  0:00.00 /usr/libexec/getty Pc ttyv3
713 v4  Is+  0:00.00 /usr/libexec/getty Pc ttyv4
714 v5  Is+  0:00.00 /usr/libexec/getty Pc ttyv5
715 v6  Is+  0:00.00 /usr/libexec/getty Pc ttyv6
716 v7  Is+  0:00.00 /usr/libexec/getty Pc ttyv7
847  0  Ss   0:00.04 -csh (csh)
883  0  R+   0:00.00 ps
```

If you also want to see processes who don't have a controlling terminal, add " `-x` " option:

```
# ps -ax
PID TT  STAT     TIME COMMAND
  0  -  DLs   0:00.10 [kernel]
  1  -  ILs   0:00.05 /sbin/init --
  2  -  DL    0:00.05 [cam]
  3  -  DL    0:00.00 [mpt_recovery0]
  4  -  DL    0:00.00 [sctp_iterator]
  5  -  DL    0:00.04 [pagedaemon]
  6  -  DL    0:00.00 [vmdaemon]
  7  -  DL    0:00.00 [pagezero]
  8  -  DL    0:00.08 [bufdaemon]
  9  -  DL    0:00.01 [vnlru]
 10  -  DL    0:00.00 [audit]
 11  -  RL   50:35.75 [idle]
 12  -  WL    0:08.20 [intr]
 13  -  DL    0:00.02 [geom]
```

```
 14  -  DL    0:00.91 [rand_harvestq]
 15  -  DL    0:00.44 [usb]
 16  -  DL    0:00.20 [syncer]
302  -  Is    0:00.01 dhclient: em0 [priv] (dhclient)
364  -  Is    0:00.00 dhclient: em0 (dhclient)
379  -  Is    0:00.00 /sbin/devd
524  -  Ss    0:00.02 /usr/sbin/syslogd -s
652  -  Is    0:00.00 /usr/sbin/sshd
655  -  Ss    0:00.10 sendmail: accepting connections (sendmail)
658  -  Is    0:00.00 sendmail: Queue runner@00:30:00 for /var/s
pool/clientmqueue (sendmail)
662  -  Ss    0:00.01 /usr/sbin/cron -s
844  -  Ss    0:00.06 sshd: root@pts/0 (sshd)
854 v0  Is+   0:00.00 /usr/libexec/getty Pc ttyv0
710 v1  Is+   0:00.00 /usr/libexec/getty Pc ttyv1
711 v2  Is+   0:00.00 /usr/libexec/getty Pc ttyv2
712 v3  Is+   0:00.00 /usr/libexec/getty Pc ttyv3
713 v4  Is+   0:00.00 /usr/libexec/getty Pc ttyv4
714 v5  Is+   0:00.00 /usr/libexec/getty Pc ttyv5
715 v6  Is+   0:00.00 /usr/libexec/getty Pc ttyv6
716 v7  Is+   0:00.00 /usr/libexec/getty Pc ttyv7
847  0  Ss    0:00.04 -csh (csh)
897  0  R+    0:00.00 ps -ax
```

This time, there are more results whose `TT` columns are `-`.

The most used "`ps`" command is "`ps auwwx`" which displays all running processes detailed info, and this is similar to "`# ps -ef`" on `GNU/Linux`:

```
 # ps auwwx
USER   PID %CPU %MEM   VSZ  RSS TT  STAT STARTED     TIME COMMAND
root    11 99.2  0.0     0   16  -  RL     5:48PM 54:14.29 [idle]
root     0  0.0  0.0     0  160  -  DLs    5:48PM  0:00.10 [kernel
]
root     1  0.0  0.0  9492  772  -  ILs    5:48PM  0:00.05 /sbin/i
nit --
root     2  0.0  0.0     0   32  -  DL     5:48PM  0:00.05 [cam]
root     3  0.0  0.0     0   16  -  DL     5:48PM  0:00.00 [mpt_re
covery0]
root     4  0.0  0.0     0   16  -  DL     5:48PM  0:00.00 [sctp_i
```

```
terator]
root     5  0.0  0.0     0   32  -  DL    5:48PM  0:00.05 [pageda
emon]
root     6  0.0  0.0     0   16  -  DL    5:48PM  0:00.00 [vmdaem
on]
root     7  0.0  0.0     0   16  -  DL    5:48PM  0:00.00 [pageze
ro]
root     8  0.0  0.0     0   32  -  DL    5:48PM  0:00.08 [bufdae
mon]
root     9  0.0  0.0     0   16  -  DL    5:48PM  0:00.01 [vnlru]
root    10  0.0  0.0     0   16  -  DL    5:48PM  0:00.00 [audit]
root    12  0.0  0.0     0  208  -  WL    5:48PM  0:08.78 [intr]
root    13  0.0  0.0     0   48  -  DL    5:48PM  0:00.02 [geom]
root    14  0.0  0.0     0   16  -  DL    5:48PM  0:00.98 [rand_h
arvestq]
root    15  0.0  0.0     0  160  -  DL    5:48PM  0:00.47 [usb]
root    16  0.0  0.0     0   16  -  DL    5:48PM  0:00.21 [syncer
]
root   302  0.0  0.1 14656 2148  -  Is    5:48PM  0:00.01 dhclien
t: em0 [priv] (dhclient)
_dhcp 364  0.0  0.1 14656 2236  -  Is    5:48PM  0:00.00 dhclien
t: em0 (dhclient)
root   379  0.0  0.2 13628 4932  -  Ss    5:48PM  0:00.00 /sbin/d
evd
root   524  0.0  0.1 14520 2092  -  Ss    5:48PM  0:00.02 /usr/sb
in/syslogd -s
root   652  0.0  0.3 61316 6564  -  Is    5:48PM  0:00.00 /usr/sb
in/sshd
root   655  0.0  0.3 24156 5380  -  Ss    5:48PM  0:00.10 sendmai
l: accepting connections (sendmail)
smmsp 658  0.0  0.2 24156 5044  -  Is    5:48PM  0:00.00 sendmai
l: Queue runner@00:30:00 for /var/spool/clientmqueue (sendmail)
root   662  0.0  0.1 16624 2212  -  Is    5:48PM  0:00.01 /usr/sb
in/cron -s
```

References:
ps -- process status;
Processes and Daemons;
Show all processes in FreeBSD.

Use "procstat" to get process info

On `FreeBSD`, you can use `procstat` command to get detailed info of process. E.g.:

```
# procstat 521
PID  PPID  PGID   SID  TSID THR LOGIN    WCHAN     EMUL
 COMM
521     1   521   521     0   1 root     select    FreeBSD ELF64
 syslogd
```

The above outputs describe the info of process whose `PID` is `521`: its `PID`, parent `PID`, command name, etc.

If you want to display all processes' info, use `-a` option:

```
# procstat -a
PID  PPID  PGID   SID  TSID THR LOGIN    WCHAN     EMUL
 COMM
0      0     0     0     0  10 -        -         -             k
ernel
1      0     1     1     0   1 root     wait      FreeBSD ELF64 i
nit
2      0     0     0     0   2 -        -         -             c
am
3      0     0     0     0   1 -        idle      -             m
pt_recovery0
4      0     0     0     0   1 -        waiting_  -             s
ctp_iterator
5      0     0     0     0   2 -        umarcl    -             p
agedaemon
6      0     0     0     0   1 -        psleep    -             v
mdaemon
7      0     0     0     0   1 -        pgzero    -             p
agezero
```

There are many options you can use for `procstat`, i.e., if you are curious about thread info, use `-t` option:

```
# procstat -t 521
PID    TID COMM             TDNAME           CPU  PRI STATE   WC
HAN
521 100065 syslogd          -                  0  120 sleep   se
lect
```

For details of every option, please refer the manual.

Sockstat command

On `FreeBSD`, we can employ `sockstat` command to list both open Internet and `UNIX` domain sockets:

```
# sockstat
USER     COMMAND    PID   FD PROTO  LOCAL ADDRESS         FOREIG
N ADDRESS
root     sshd       45768 3  tcp4   192.168.80.129:22     192.16
8.80.1:60803
root     sshd       22144 3  tcp4   192.168.80.129:22     192.16
8.80.1:54834
root     sshd       19097 3  tcp4   192.168.80.129:22     192.16
8.80.1:58174
smmsp    sendmail   662   3  dgram  -> /var/run/log
root     sendmail   659   3  tcp4   127.0.0.1:25          *:*
root     sendmail   659   4  dgram  -> /var/run/logpriv
root     sshd       656   3  tcp6   *:22                  *:*
root     sshd       656   4  tcp4   *:22                  *:*
......
```

We can see the output of `sockstat` lists which process owns the socket, the file descriptor number of the socket, etc. So it is a really awesome tool to diagnose network related issues.

Reference:
SOCKSTAT(1).

Use "freecolor" to display memory and swap usage

`Freecolor` on `FreeBSD` is equal to `free` on `GNU/Linux`. Install it:

```
# cd /usr/ports/sysutils/freecolor
# make install clean
```

Display the memory and swap usage in bar format:

```
# freecolor
Physical  : [###################################..] 94%   (1907820
/2018396)
Swap      : [#####################################] 100%  (1048540
/1048540)
```

Display usage in traditional "`free`" format:

```
# freecolor -m -o
             total       used       free     shared    buffers
   cached
Mem:          1971        107       1863          0          0
        0
Swap:         1023          0       1023
```

Reference:
FreeBSD find out RAM size Including Total Amount of Free and Used Memory Size.

Notices of using Subversion

If you want to use `Subversion`, for example, updating ports tree, you should pay attention to the following issues:

(1) If your server works behind a proxy, you should modify your `Subversion` configuration file:

```
# cat ~/.subversion/servers
[global]
http-proxy-host = web-proxy.xxxx.com
http-proxy-port = 8080
http-compression = no
```

Otherwise, you may encounter following errors:

```
# svn checkout https://svn.FreeBSD.org/ports/head /usr/ports
Error validating server certificate for 'https://svn.freebsd.org
:443':
 - The certificate is not issued by a trusted authority. Use the
   fingerprint to validate the certificate manually!
 - The certificate hostname does not match.
Certificate information:
 - Hostname: FG3K6C3A15800021
 - Valid: from Mar 21 09:15:26 2015 GMT until Mar 21 09:15:26 20
25 GMT
 - Issuer: FG3K6C3A15800021, Fortinet Ltd.
 - Fingerprint: 1D:F4:21:20:2F:06:41:45:3F:7A:18:FB:79:F3:BB:30:
36:32:22:A3
(R)eject, accept (t)emporarily or accept (p)ermanently? p
svn: E170013: Unable to connect to a repository at URL 'https://
svn.freebsd.org/ports/head'
svn: E000054: Error running context: Connection reset by peer
```

Or:

```
# svn checkout https://svn.FreeBSD.org/ports/head /usr/ports
......
svn: E175002: REPORT request on '/ports/!svn/me' failed
```

(2) If you meet similar errors like as follows executing `svn` command:

```
svn: E200030: SQLite compiled for 3.11.1, but running with 3.9.2
```

You should update your `SQLite` :

```
# pkg upgrade sqlite3
```

References:
Using FreeBSD inside a controlled network – A required HTTP Proxy and No FTP;
How to configure a HTTP proxy for svn.

Print PCI devices info

`FreeBSD` provides `pciconf` command which can list `PCI` devices info:

```
# pciconf -lv
hostb0@pci0:0:0:0:  class=0x060000 card=0x197615ad chip=0x719080
86 rev=0x01 hdr=0x00
vendor = 'Intel Corporation'
device = '440BX/ZX/DX - 82443BX/ZX/DX Host bridge'
class  = bridge
subclass   = HOST-PCI
pcib1@pci0:0:1:0:   class=0x060400 card=0x00000000 chip=0x719180
86 rev=0x01 hdr=0x01
vendor = 'Intel Corporation'
device = '440BX/ZX/DX - 82443BX/ZX/DX AGP bridge'
class  = bridge
subclass   = PCI-PCI
isab0@pci0:0:7:0:   class=0x060100 card=0x197615ad chip=0x711080
86 rev=0x08 hdr=0x00
vendor = 'Intel Corporation'
device = '82371AB/EB/MB PIIX4 ISA'
class  = bridge
subclass   = PCI-ISA
......
```

If you are accustomed to `lspci` which is shipped on `GNU/Linux`, you can install it yourself:

```
# pkg install pciutils
......
# lspci
00:00.0 Host bridge: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/
DX Host bridge (rev 01)
00:01.0 PCI bridge: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/D
X AGP bridge (rev 01)
00:07.0 ISA bridge: Intel Corporation 82371AB/EB/MB PIIX4 ISA (r
ev 08)
......
```

Reference:
"lspci" & pciconf commands in FreeBSD.

Mount procfs file system

`FreeBSD` doesn't mount `procfs` file system by default, if you need it, you can mount it yourself:

```
mount -t procfs proc    /proc
```

If you want the `procfs` is automatically mounted at boot time, you can add the following line in `/etc/fstab` file:

```
proc         /proc    procfs   rw    0 0
```

References:
procfs: Gone But Not Forgotten;
procfs -- process file system.

Kldstat command

On `FreeBSD`, you can use `kldstat` command to display the status of any files dynamically linked into the kernel:

```
# kldstat
Id Refs Address            Size     Name
 1    6 0xffffffff80200000 17bc6a8  kernel
 2    1 0xffffffff81a11000 56c6     fdescfs.ko
 3    1 0xffffffff81a17000 2ba8     uhid.ko
```

If you want more detailed info, you can use `-v` option:

```
# kldstat -v
Id Refs Address            Size     Name
 1    6 0xffffffff80200000 17bc6a8  kernel (/boot/kernel/kernel)
 Contains modules:
  Id Name
  386 if_lo
  398 newreno
  373 elf32
  372 elf64
  374 shell
.....
```

Reference:
KLDSTAT(8)).

Load and unload files into kernel

You can use `kldload` command to load files into kernel. For example, enter `/boot/kernel` directory, and use `kldstat` to check currently loaded files:

```
# cd /boot/kernel
# kldstat
Id Refs Address            Size     Name
 1    6 0xffffffff80200000 17bc6a8  kernel
 2    1 0xffffffff81a11000 56c6     fdescfs.ko
 3    1 0xffffffff81a17000 2ba8     uhid.ko
```

Use `kldload` to load `zfs.ko` file:

```
# kldload zfs.ko
# kldstat
Id Refs Address            Size     Name
 1   15 0xffffffff80200000 17bc6a8  kernel
 2    1 0xffffffff81a11000 56c6     fdescfs.ko
 3    1 0xffffffff81a17000 2ba8     uhid.ko
 7    1 0xffffffff81a1a000 1ee0c8   zfs.ko
 8    1 0xffffffff81c09000 3330     opensolaris.ko
```

You can see the `zfs.ko` is loaded successfully.

Unloading file uses `kldunload` command:

```
# kldunload zfs.ko
# kldstat
Id Refs Address            Size     Name
 1    6 0xffffffff80200000 17bc6a8  kernel
 2    1 0xffffffff81a11000 56c6     fdescfs.ko
 3    1 0xffffffff81a17000 2ba8     uhid.ko
```

Reference:
kldload;
kldunload.